SPRING IOC CONTAINERS

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction. The Spring container uses dependency injection (DI) to manage the components that make up an application. These objects are called Spring Beans which we will discuss in next chapter.

The container gets its instructions on what objects to instantiate, configure, and assemble by reading configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram is a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.
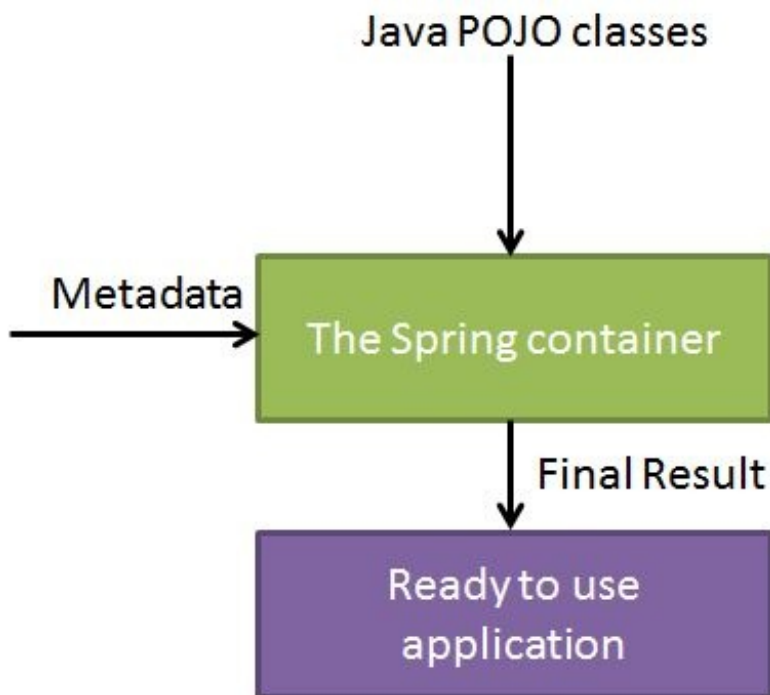
Spring provides following two distinct types of containers.

S.N.	Container & Description
1	**Spring BeanFactory Container** This is the simplest container providing basic support for DI and defined by the *org.springframework.beans.factory.BeanFactory* interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purposes of backward compatibility with the large number of third-party frameworks that integrate with Spring.
2	**Spring ApplicationContext Container** This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the *org.springframework.context.ApplicationContext* interface.

The *ApplicationContext* container includes all functionality of the *BeanFactory* container, so it is generally

recommended over the *BeanFactory*. BeanFactory can still be used for light weight applications like mobile devices or applet based applications where data volume and speed is significant.